

# Safety of Deferred Update in Transactional Memory \*

Hagit Attiya<sup>1</sup>   Sandeep Hans<sup>1</sup>   Petr Kuznetsov<sup>2</sup>   Srivatsan Ravi<sup>2</sup>

<sup>1</sup>Department of Computer Science, Technion

<sup>2</sup>Telekom Innovation Laboratories, TU Berlin

## Abstract

Transactional memory allows the user to declare sequences of instructions as speculative *transactions* that can either *commit* or *abort*. If a transaction commits, its effect appears executed sequentially, so that the committed transactions constitute a correct sequential execution. If a transaction aborts, none of its update instructions can affect other transactions.

The popular criterion of *opacity* requires that the views of aborted transactions must also be consistent with the global sequential order constituted by committed ones. This is believed to be important, since inconsistencies observed by an aborted transaction may cause a fatal irrecoverable error or waste of the system in an infinite loop. Intuitively, an opaque implementation must ensure that no intermediate view a transaction obtains before it commits or aborts can be affected by a transaction that has not started committing yet, so called *deferred-update* semantics.

In this paper, we intend to grasp this intuition formally. We propose a variant of opacity that explicitly requires the sequential order to respect the deferred-update semantics. We show that our criterion is a safety property, i.e., it is prefix- and limit-closed. Unlike opacity, our property also ensures that a serialization of a history implies serializations of its prefixes maintaining original “read-from” relations. Finally, we show that our property is equivalent to opacity if we assume that no two transactions commit identical values on the same variable, and present a counter-example for scenarios when the “unique-write” assumption does not hold.

## 1 Introduction

Resolving conflicts in an efficient and consistent manner is the most challenging task in concurrent software design. Transactional memory (TM) [7, 14] addresses this challenge by offering an interface in which sequences of shared-memory instructions can be declared as speculative *transactions*. The underlying idea, borrowed from databases, is to treat each transaction as an atomic event: a transaction may either *commit* in which case it appears as executed sequentially, or *abort* in which case none of its update instructions affect other transactions. The user can therefore design software having only sequential semantics in mind and let the memory take care of conflicts resulting from potentially concurrent executions.

---

\*The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement N 238639, ITN project TRANSFORM, and grant agreement N 248465, the S(o)S project.

In databases, a correct implementation of concurrency control should guarantee that committed transactions constitute a serial (or sequential) execution [6]. On the other hand, uncommitted transactions can be aborted without invalidating the correctness of committed ones. (In the literature on databases, the latter feature is called *recoverability*.)

In the TM context, intermediate states witnessed by an incomplete transaction may affect the application through the outcome of its read operations. If the intermediate state is not consistent with any sequential execution, the application may experience a fatal irrecoverable error or sink in an infinite loop. The correctness criterion of *opacity* [4, 5] addresses this issue by requiring the states observed by uncommitted transactions to be consistent with a global serial execution constituted by committed ones (a *serialization*).

An opaque TM implementation must, intuitively, ensure that no transaction can read from a transaction that has not started committing yet. This is usually referred to as the *deferred-update* semantics, and it was in fact explicitly required in some representations of opacity [3]. The motivation of this paper is to capture this intuition formally.

We present a new correctness criterion called *du-opacity*. Our criterion defines the *read-from* relation for each read operation performed in a serial TM execution. Informally, a du-opaque (possibly, non-serial) execution must be indistinguishable from a totally-ordered execution, with respect to which no transaction reads from a transaction that has not started committing.

We further show that our correctness criterion is a *safety property*, as defined by Owicky and Lamport [1, 13], Alpern and Schneider [1] and refined by Lynch [12]. We show that du-opacity is *prefix-closed*: every prefix of a du-opaque history is also du-opaque. We also show that du-opacity is, under certain restrictions, *limit-closed*. More precisely, assuming that, in an infinite execution, every transaction completes (commits or aborts), the infinite limit of any sequence of ever extending du-opaque histories is also du-opaque. To prove du-opacity for such an implementation, it is thus sufficient to prove that all its *finite* histories are du-opaque. To the best of our knowledge, this paper contains the first non-trivial proof of limit-closure for a TM correctness property. We further show that any du-opaque serialization of a history implies a serialization of any of its prefixes that maintains the original read-from relations, which is instrumental in the comparison of du-opacity with opacity.

Opacity, as defined in [5], reduces correctness of an infinite history to correctness of all its prefixes, and thus is limit-closed by definition. In fact, we show that extending opacity to infinite histories in a non-trivial way (i.e., requiring that even infinite histories should have proper serializations), does not result in a limit-closed property. We observe that opacity does not preclude scenarios in which a transaction reads from a future transaction (cf. examples in Figures 4 and 5), and, thus, our criterion is strictly stronger than opacity. Surprisingly, this is true even if we assume that all transactional operations are atomic, which somewhat attenuates earlier attempts to forcefully introduce the deferred-update in the definition of opacity for atomic operations [3]. However, we show that opacity and du-opacity are equivalent if we assume that no two transactions try to commit identical values on the same data item.

We believe that these results improve our understanding of the very notion of correctness in transactional memory. Our correctness criterion explicitly declares that a transaction is not allowed to read from an uncommitted transaction, and we conjecture that it is simpler to verify. We present the first non-trivial proof for both limit- and prefix-closure of TM histories, which is quite interesting in its own right, for it enables reasoning about possible serializations of an infinite TM history based on serializations of its prefixes.

The paper is organized as follows. In Section 2, we introduce our basic model definitions and recall the notion of safety [1, 12, 13]. In Section 3, we define our criterion of du-opacity and show that it is, under certain restrictions, a safety property. In Section 4, we prove that du-opacity is a proper subset of the original notion of opacity [5], and that it coincides with

du-opacity under the “unique-writes” condition.

## 2 Model

A *Transactional Memory* (in short, *TM*) supports atomic *transactions* for reading and writing a set of *transactional* objects (in short, *t-objects*). A transaction is a sequence of accesses (reads or writes) to t-objects; each transaction  $T_k$  has a unique identifier  $k$ .

A transaction  $T_k$  may contain the following *t-operations*, each being a pair of *invocation* and *response* events:

1.  $read_k(X)$  returns a value in some domain  $V$  or a special value  $A_k \notin V$  (*abort*);
2.  $write_k(X, v)$ , for a value  $v \in V$ , returns  $ok_k$  or  $A_k$ ;
3.  $tryC_k$  returns  $C_k \notin V$  (*commit*) or  $A_k$ ; and
4.  $tryA_k$  returns  $A_k$ .

The *read set* (resp., the *write set*) of a transaction  $T_k$ , denoted  $Rset(T_k)$ , is the set of t-objects that  $T_k$  reads in  $H$ ; the *write set* of  $T_k$ , denoted  $Wset(T_k)$ , is the set of t-objects  $T_k$  writes to in  $H$ . The *data set* of  $T_k$  is  $Dset(T_k) = Rset(T_k) \cup Wset(T_k)$ .

We consider an asynchronous shared-memory system in which processes communicate via transactions. A *TM implementation* provides processes with algorithms for implementing  $read_k$ ,  $write_k$ ,  $tryC_k()$  and  $tryA_k()$  of a transaction  $T_k$ .

A *history* of a TM implementation is a (possibly infinite) sequence of invocation and response *events* of t-operations.

For every transaction identifier  $k$ ,  $H|k$  denotes the subsequence of  $H$  restricted to events of transaction  $T_k$ . If  $H|k$  is non-empty, we say that  $T_k$  *participates* in  $H$ , and let  $txns(H)$  denote the set of transactions that participate in  $H$ .

Two histories  $H$  and  $H'$  are *equivalent* if  $txns(H) = txns(H')$  and for every transaction  $T_k \in txns(H)$ ,  $H|k = H'|k$ .

A history  $H$  is *sequential* if every invocation of a t-operation is either the last event in  $H$  or is immediately followed by a matching response. A history is *well-formed* if for all  $T_k$ ,  $H|k$  is sequential and has no events after  $A_k$  or  $C_k$ . We assume that all histories are well-formed, i.e., the client of the transactional memory never invokes a t-operation before receiving a response from the previous one and does not invoke any t-operation  $op_k$  after receiving  $C_k$  or  $A_k$ .

A transaction  $T_k \in txns(H)$  is *complete in  $H$*  if  $H|k$  ends with a response event. The history  $H$  is *complete* if all transactions in  $txns(H)$  are complete in  $H$ .

A transaction  $T_k \in txns(H)$  is *t-complete* if  $H|k$  ends with  $A_k$  or  $C_k$ ; otherwise,  $T_k$  is *t-incomplete*.  $T_k$  is *committed* (resp., *aborted*) in  $H$  if the last event of  $T_k$  is  $C_k$  (resp.,  $A_k$ ). The history  $H$  is *t-complete* if all transactions in  $txns(H)$  are t-complete.

For t-operations  $op_k, op_j$ , we say that  $op_k$  *precedes*  $op_j$  in the *real-time order* of  $H$ , denoted  $op_k \prec_H^{RT} op_j$ , if the response of  $op_k$  precedes the invocation of  $op_j$ .

For transactions  $T_k, T_m \in txns(H)$ , we say that  $T_k$  *precedes*  $T_m$  in the *real-time order* of  $H$ , denoted  $T_k \prec_H^{RT} T_m$ , if  $T_k$  is t-complete in  $H$  and the last event of  $T_k$  precedes the first event of  $T_m$  in  $H$ . If neither  $T_k \prec_H^{RT} T_m$  nor  $T_m \prec_H^{RT} T_k$ , then  $T_k$  and  $T_m$  *overlap* in  $H$ .

A history  $H$  is *t-sequential* if there are no overlapping transactions in  $H$ .

For simplicity of presentation, we assume that each history  $H$  begins with an “imaginary” transaction  $T_0$  that writes initial values to all t-objects and commits before any other transaction begins in  $H$ .

Let  $H$  be a t-sequential history. For every operation  $read_k(X)$  in  $H$ , we define the *latest written value* of  $X$  as follows:

1. If  $T_k$  contains a  $write_k(X, v)$  preceding  $read_k(X)$ , then the latest written value of  $X$  is value of the latest such write to  $X$ .
2. Otherwise, if  $H$  contains a  $write_m(X, v)$ ,  $T_m$  precedes  $T_k$ , and  $T_m$  commits in  $H$ , then the latest written value of  $X$  is the value of the latest such write to  $X$  in  $H$ . (This write is well-defined since  $H$  starts with  $T_0$  writing to all t-objects.)

A t-complete t-sequential history  $S$  is *legal* if every  $read_k(X)$  in  $H$  that does not return  $A_k$ , returns the latest written value of  $X$ .

**Definition 1.** A property  $\mathcal{P}$  is a set of (transactional) histories. A property  $\mathcal{P}$  is a safety property if it satisfies the following two conditions [1, 12]:

1. Prefix-closure: every prefix  $H'$  of a history  $H \in \mathcal{P}$  is also in  $\mathcal{P}$  and
2. Limit-closure: for any infinite sequence of finite histories  $H^0, H^1, \dots$  such that for all  $i$ ,  $H^i \in \mathcal{P}$  and  $H^i$  is a prefix of  $H^{i+1}$ , the infinite history that is the limit of the sequence is also in  $\mathcal{P}$ .

Notice that the set of histories produced by a TM implementation  $M$  is prefix-closed. Therefore, every infinite history of  $M$  is the limit of an infinite sequence of ever-extending finite histories of  $M$ . Thus, to prove that  $M$  satisfies a safety property  $P$ , it is enough to show that all finite histories of  $M$  are in  $P$ . Indeed, limit-closure of  $P$  then implies that every infinite history of  $M$  is also in  $P$ .

### 3 DU-Opacity

In this section, we introduce our correctness criterion, *du-opacity*, and prove that a restriction of it is a safety property.

**Definition 2.** Let  $H$  be any history. A completion of  $H$ , denoted  $\bar{H}$ , is a history derived from  $H$  as follows:

- for every incomplete t-operation  $op_k$  of  $T_k \in \text{txns}(H)$  in  $H$ , if  $op_k = read_k \vee write_k$ , insert  $A_k$  somewhere after the invocation of  $op_k$ ; else if  $op_k = tryA_k() \vee tryC_k()$ , insert a matching response to  $op_k$  somewhere after the invocation of  $op_k$ .
- for every complete transaction  $T_k \in \text{txns}(H)$ , insert  $tryC_k \cdot A_k$  somewhere after the last event of transaction  $T_k$ .

Now we define our correctness criterion. We begin with defining what it means for a transaction to read from another transaction in a t-sequential legal history.

Let  $S$  be a legal t-sequential history, and let  $<_S$  be the total order on t-operations in  $S$ .

Consider a read operation  $read_k(X)$  in  $S$  that does not abort, i.e., which returns a value  $v \in V$ ; its *read-from transaction*, denoted  $\rho_S(read_k(X))$ , is defined as follows:

- (1) If there is  $write_k(X, v)$  performed by  $T_k$  that is the latest write in  $T_k$  such that  $write_k(X, v) <_S read_k(X)$ , then  $\rho_S(read_k(X)) = T_k$ . I.e., if the same transaction writes to  $X$ , then  $read_k(X)$  is mapped to  $T_k$ .

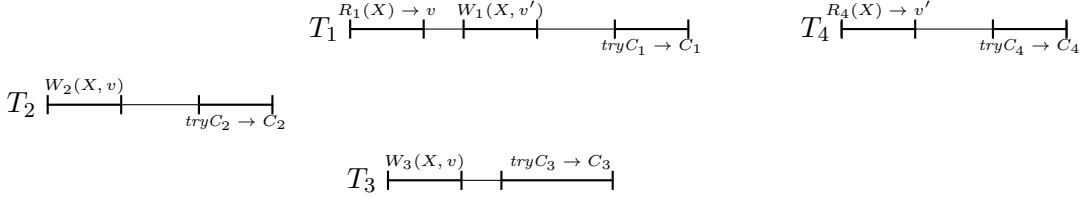


Figure 1: A du-opaque history  $H$ ; for any serialization  $S$  of  $H$ ,  $\rho_S(R_1(X) = T_2)$  and  $\rho_S(R_4(X) = T_1)$

- (2) Otherwise,  $\rho_S(\text{read}_k(X)) = T_m$ , where  $T_m$  is the earliest committed transaction in  $S$  that performs  $\text{write}_m(X, v)$  such that  $T_m <_S T_k$ , and there is no committed transaction  $T_i$  in  $S$  that performs  $\text{write}_i(X, v')$ ,  $v' \neq v$ , such that  $T_m <_S T_i <_S T_k$ . I.e.,  $\text{read}_k(X)$  is mapped to the earliest committed transaction writing the value it read, which is not overwritten.

**Definition 3.** A history  $H$  is du-opaque if there is a legal  $t$ -complete  $t$ -sequential history  $S$  such that

- (1) for every pair of transactions  $T_k, T_m \in \text{tns}(H)$ , if  $T_k \prec_H^{RT} T_m$ , then  $T_k <_S T_m$ , i.e.,  $S$  respects the real-time ordering of transactions in  $H$ , and
- (2) there exists a completion of  $H$  that is equivalent to  $S$ , and
- (3) for each  $\text{read}_k(X)$  in  $H$  that does not return  $A_k$  such that  $\rho_S(\text{read}_k(X)) = T_m$ ;  $\text{read}_k(X) \not\prec_H^{RT} \text{tryC}_m()$ .

We then say that  $S$  is a (du-opaque) serialization of  $H$ . Let  $\text{seq}(S)$  denote the sequence of transactions in  $S$  and  $\text{seq}(S)[k]$  be the  $k^{\text{th}}$  transaction in this sequence.

Figure 1 presents a du-opaque history  $H$  to illustrate the usage of the notion of a read-from transaction. Let  $S$  be the  $t$ -complete  $t$ -sequential history such that  $\text{seq}(S) = T_2, T_3, T_1, T_4$  and  $S$  is equivalent to  $H$ . It is easy to see that  $S$  is legal and respects the real-time order of  $H$ . By definition,  $\rho_S(\text{read}_1(X)) = T_2$  since  $T_2$  is the earliest committed transaction in  $S$  that writes  $v$  to  $X$ . Then we observe that  $\text{read}_1(X) \not\prec_H^{RT} \text{tryC}_2()$ . For  $\text{read}_4(X)$ ,  $\rho_S(\text{read}_4(X)) = T_1$ . Further,  $\rho_S(\text{read}_4(X)) = T_1$  and  $\text{read}_4(X) \not\prec_H^{RT} \text{tryC}_1()$ . Thus,  $S$  is a du-opaque serialization of  $H$ .

Now we show that du-opacity has an important property: every serialization of a du-opaque history yields a serialization for each of its prefixes that preserves the read-from relations of the original history intact. For a history  $H$ , let  $H^i$  be the finite prefix of  $H$  of length  $i$  (consisting of the first  $i$  events of  $H$ ).

**Theorem 1.** Let  $H$  be a du-opaque history and  $S$  be a du-opaque serialization of  $H$ . For any  $i \in \mathbb{N}$ , there is a serialization  $S^i$  of  $H^i$ , such that (1)  $\text{seq}(S^i)$  is a subsequence of  $\text{seq}(S)$  and (2) for every  $\text{read}_k(X)$  in  $H^i$  that does not return  $A_k$ , if  $\rho_S(\text{read}_k(X)) = T_m$ , then  $\rho_{S^i}(\text{read}_k(X)) = T_m$ .

*Proof.* Given  $H$ ,  $S$  and  $H^i$ , define a  $t$ -complete  $t$ -sequential history  $S^i$  as follows:

- for every  $t$ -complete transaction  $T_k$  in  $H^i$ ,  $S^i|k = S|k$ .
- for every complete transaction  $T_k$  in  $H^i$  that is not  $t$ -complete,  $S^i|k$  consists of the sequence of events in  $H^i|k$ , immediately followed by  $\text{tryC}_k() \cdot A_k$ .

- for every transaction  $T_k \in \text{txns}(H^i)$  with an incomplete t-operation  $op_k = read_k \vee write_k$  in  $H^i$ ,  $S^i|k$  is the sequence of events in  $S|k$  up to the invocation of  $op_k$ , immediately followed by  $A_k$ .
- for every transaction  $T_k \in \text{txns}(H^i)$  with an incomplete t-operation  $op_k = tryC_k() \vee tryA_k()$ ,  $S^i|k = S|k$ .

By construction,  $\text{txns}(S^i) \subseteq \text{txns}(S)$ . We further require that  $seq(S^i)$  is a subsequence of  $seq(S)$ .

Since  $S^i$  is derived from events in  $\bar{H}$  (some completion of  $H$  that is equivalent to  $S$ ), there is a completion of  $H^i$  that is equivalent to  $S^i$ , since  $S^i$  contains events from every complete t-operation in  $H^i$  and other events included clearly conform to Definition 2.

We now claim that  $S^i$  is a serialization of  $H^i$ .

- (1) Suppose, by way of contradiction, that for some pair of transactions  $T_j, T_k \in \text{txns}(H^i)$ ,  $T_j \prec_{H^i}^{RT} T_k$ , but  $T_j \not\prec_{S^i} T_k$ . But this implies that  $T_j \not\prec_S T_k$  since  $T_j, T_k \in \text{txns}(H)$ —contradiction.
- (2) Suppose, by way of contradiction, that  $S^i$  is not legal, i.e., there is some  $read_k(X)$  in  $H^i$  that does not return the latest written value of  $X$  in  $S^i$ . There are two cases:
  - Suppose that there is  $write_k(X, v)$  performed by  $T_k$  that is the latest write in  $T_k$  such that  $write_k(X, v) \prec_{H^i}^{RT} read_k(X)$ . Then, if  $v$  is not the latest written value of  $X$  in  $S^i$ , it is also not the latest written value of  $X$  in  $S$ , which is a contradiction.
  - Suppose that there is no  $write_k(X, v)$  performed by  $T_k$ .  $S$  is a serialization of  $H$ , thus we can consider transaction  $T_m = \rho_S(read_k(X))$ , such that  $read_k(X) \not\prec_{H^i}^{RT} tryC_m()$ . Hence,  $T_m \in \text{txns}(H^i)$ . By construction of  $S^i$ ,  $T_m \in \text{txns}(S^i)$  and  $T_m$  is committed in  $S^i$ . By assumption,  $read_k(X)$  returns  $v$  in  $H^i$  such that  $v$  is not the latest written value of  $X$  in  $S^i$ . Hence, there exists a committed transaction  $T_j$  that performs  $write_j(X, v')$ ;  $v' \neq v$  in  $S^i$  such that  $T_m <_{S^i} T_j <_{S^i} T_k$ . But this is not possible since  $seq(S^i)$  is a subsequence of  $seq(S)$ , contradicting the fact that  $T_m = \rho_S(read_k(X))$ .

Thus,  $S^i$  is a legal t-complete t-sequential history equivalent to some completion of  $H^i$ .

- (3) Suppose, by way of contradiction, that there is a  $read_k(X)$  in  $H^i$  such that  $\rho_S(read_k(X)) = T_m$ , but  $\rho_{S^i}(read_k(X)) \neq T_m$ .  $T_m \in \text{txns}(H^i)$  since  $read_k(X) \not\prec_{H^i}^{RT} tryC_m()$ . By construction of  $S^i$ ,  $T_m \in \text{txns}(S^i)$  and  $T_m$  is committed in  $S^i$ . The only possibility is that there exists a committed transaction  $T_j \in \text{txns}(H^i)$  that performs  $write_j(X, v)$  such that  $T_j <_{S^i} T_m$ . However, this implies that  $T_j <_S T_m$  i.e.  $T_m$  is not the read-from transaction for  $read_k(X)$  in  $S$ , which is contradiction.
- (4) The above argument implies that for every  $read_k(X)$  in  $H^i$ , if  $\rho_{S^i}(read_k(X)) = T_m$ , then  $read_k(X) \not\prec_{H^i}^{RT} tryC_m()$ .

□

Theorem 1 immediately implies that du-opacity is prefix closed.

**Corollary 2.** *DU-Opacity is a prefix-closed property.*

We show now that, in general, du-opacity is not a limit-closed property by presenting an infinite history that is not du-opaque, but every prefix of it is du-opaque.

**Proposition 1.** *DU-Opacity is not a limit-closed property.*

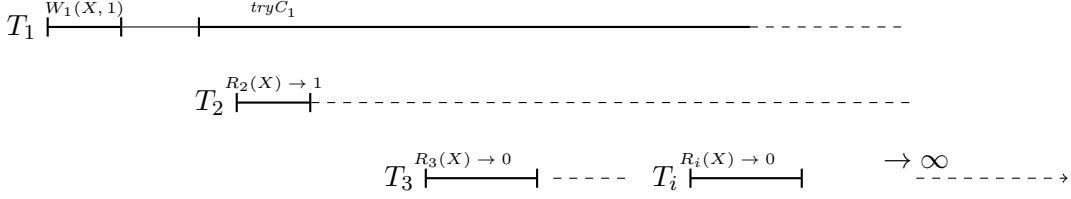


Figure 2: Each finite prefix of the history is du-opaque, but the infinite limit of the ever-extending sequence is not du-opaque

*Proof.* Let  $H^j$  denote a finite prefix of  $H$  of length  $j$ . Consider the infinite limit history  $H$  that is the limit of the histories  $H^j$  defined as follows (see Figure 2):

- Transaction  $T_1$  performs a  $write_1(X, 1)$  and then invokes  $tryC_1()$  that is incomplete in  $H$ .
- Transaction  $T_2$  performs a  $read_2(X)$  that overlaps with  $tryC_1()$  and returns 1.
- There are infinitely many transactions  $T_i, i \geq 3$ , each of which performing a single  $read_i(X)$  that returns 0 such that each  $T_i$  overlaps with both  $T_1$  and  $T_2$ .

A t-complete t-sequential history  $S^j$  is derived from the sequence  $T_3, \dots, T_j, T_0, T_1$  in which (1)  $tryC_1()$  is completed by inserting  $C_1$  immediately after its invocation and (2) any incomplete  $read_j(X)$  is completed by inserting  $A_j$  immediately after its invocation. It is easy to observe that  $S^j$  is indeed a serialization of  $H^j$ .

However, there is no serialization of  $H$ . Suppose that such a serialization  $S$  exists. Since every transaction that participates in  $H$  must participate in  $S$ , there exists  $n \in \mathbb{N}$  such that  $seq(S)[n] = T_1$ . Consider the transaction at index  $n + 1$ , say  $T_i$  in  $seq(S)$ . But for any  $i \geq 3$ ,  $T_i$  must precede  $T_1$  in any serialization (by legality), which is a contradiction.  $\square$

We next prove that du-opacity is limit-closed if we assume that, in an infinite history, every transaction eventually commits or aborts.

The proof uses König's Path Lemma on a rooted directed graph,  $G$ . Let  $v_0$  be the root vertex of  $G$ ; a vertex of  $G$ ,  $v_k$ , is *reachable* from  $v_0$ , if there is a sequence of vertices  $v_0 \dots, v_k$  such that for each  $i$ , there exists an edge from  $v_i$  to  $v_{i+1}$ .  $G$  is *connected* if every vertex in  $G$  is reachable from  $v_0$ .  $G$  is *finitely branching* if every vertex in  $G$  has a finite out-degree.  $G$  is *infinite* if the set of vertices in  $G$  is infinite.

**Lemma 3** (König's Path Lemma [9]). *If  $G$  is an infinite connected finitely branching rooted directed graph, then  $G$  contains an infinite sequence of vertices  $v_0, v_1, \dots$  such that  $v_0$  is the root, for every  $i \geq 0$ , there is an edge from  $v_i$  to  $v_{i+1}$ , and for every  $i \neq j$ ,  $v_i \neq v_j$ .*

**Theorem 4.** *Under the restriction that in any infinite history  $H$ , every transaction  $T_k \in \text{trans}(H)$  is t-complete, du-opacity is a limit-closed property.*

*Proof.* We are given an infinite sequence of finite histories  $H^0, H^1, \dots, H^i, H^{i+1}, \dots$  such that for all  $i$ ,  $H^i$  is a prefix of  $H^{i+1}$ . Let  $H$  be the corresponding infinite limit history. Assume that for all  $i$ ,  $H^i$  is du-opaque; we prove that  $H$  is also du-opaque.

We construct a rooted directed graph  $G_H$  for  $H$ .

- (0) The root vertex of  $G_H$  is  $(H^0, S^0)$  where  $S^0$  and  $H^0$  contain the initial transaction  $T_0$ .
- (1) Each non-root vertex of  $G_H$  is a tuple  $(H^i, S^i_\alpha)$ , where  $S^i_\alpha$  is some du-opaque serialization of  $H^i$ . In the rest of this proof, when we refer to a specific  $S^i$ , it is understood to be associated with the prefix  $H^i$  of  $H$ .

- (2) Let  $cseq_i(S^j)$ ,  $j \geq i$ , denote the subsequence of  $seq(S^j)$  reduced to  $t$ -complete transactions in  $H^i$ . For every pair of vertices  $v = (H^i, S^i)$  and  $v' = (H^{i+1}, S^{i+1})$  in  $G_H$ , there is an edge from  $v$  to  $v'$  if  $cseq_i(S^i) = cseq_i(S^{i+1})$ .

The out-degree of a vertex  $v = (H^i, S^i)$  in  $G_H$  is defined by the cardinality of the set of the possible serializations of  $H^{i+1}$ . The out-degree is bounded by the number of possible permutations of the set  $txns(S^{i+1})$ , implying that  $G_H$  is *finitely branching*.

By Theorem 1, given a serialization  $S^{i+1}$  of  $H^{i+1}$ , there exists a serialization  $S^i$  of  $H^i$  such that  $seq(S^i)$  is a subsequence of  $seq(S^{i+1})$ . Since  $seq(S^{i+1})$  contains every  $t$ -complete transaction in  $H^i$ ,  $cseq_i(S^i) = cseq_i(S^{i+1})$ . Therefore, for every vertex  $(H^{i+1}, S^{i+1})$ , there is a vertex  $(H^i, S^i)$  such that  $cseq_i(S^i) = cseq_i(S^{i+1})$ . Therefore, we can iteratively construct a path from  $(H^0, S^0)$  to every vertex  $(H^i, S^i)$  in  $G_H$ , implying that  $G_H$  is *connected*.

We now apply König's Path Lemma to  $G_H$ . Since  $G_H$  is an infinite connected finitely branching rooted directed graph, we can derive an infinite sequence of non-repeating vertices

$$\mathcal{L} = (H^0, S^0), (H^1, S^1), \dots, (H^i, S^i), \dots$$

such that  $cseq_i(S^i) = cseq_i(S^{i+1})$ .

The rest of the proof explains how to use  $\mathcal{L}$  to construct a serialization of  $H$ . We begin with the following claim concerning  $\mathcal{L}$ .

**Claim 5.** *For any  $j > i$ ,  $cseq_i(S^i) = cseq_i(S^j)$ .*

*Proof.* From  $\mathcal{L}$ , we have the following relations:  $cseq_i(S^i)$  is a prefix of  $cseq_i(S^{i+1})$ , and  $cseq_{i+1}(S^{i+1})$  is a prefix of  $cseq_{i+1}(S^{i+2})$ . By definition,  $cseq_i(S^{i+1})$  is a subsequence of  $cseq_{i+1}(S^{i+1})$  since every transaction that is  $t$ -complete in  $H^{i+1}$  is also  $t$ -complete in  $H^i$ . Hence,  $cseq_i(S^i)$  is a subsequence of  $cseq_{i+1}(S^{i+2})$ . But,  $cseq_{i+1}(S^{i+2})$  is a subsequence of  $cseq_{i+2}(S^{i+2})$ . Thus,  $cseq_i(S^i)$  is a subsequence of  $cseq_{i+2}(S^{i+2})$ . This argument can be extended to show that for any  $j > i$ ,  $cseq_i(S^i)$  is a subsequence of  $cseq_j(S^j)$ . By definition,  $cseq_i(S^j)$  is the subsequence of  $cseq_j(S^j)$  reduced to  $t$ -complete transactions in  $H^i$ . Thus,  $cseq_i(S^i)$  is indeed equal to  $cseq_i(S^j)$ .  $\square$

Let  $f : \mathbb{N} \rightarrow txns(H)$  be defined as follows:  $f(1) = T_0$ . For every integer  $k > 1$ , let

$$i_k = \min\{\ell \in \mathbb{N} \mid \forall j > \ell : cseq_\ell(S^\ell)[k] = cseq_j(S^j)[k]\}$$

Then,  $f(k) = cseq_{i_k}(S^{i_k})[k]$ .

**Claim 6.** *The function  $f$  is total and bijective.*

*Proof. (Totality)* Since each transaction  $T \in txns(H)$  is  $t$ -complete in some prefix  $H^i$  of  $H$ , there are  $i, k \in \mathbb{N}$  such that  $cseq_i(S^i)[k] = T$ . By Claim 5, for any  $j > i$ ,  $cseq_i(S^i) = cseq_i(S^j)$ . Since every transaction that is  $t$ -complete in  $H^i$  is also  $t$ -complete in  $H^j$ , it follows that for every  $j > i$ ,  $cseq_j(S^j)[k'] = T$ , with  $k' \geq k$ .

Since  $T$  is  $t$ -complete in  $H$ , there is  $i$ ,  $cseq_i(S^i)[k] = T$ , such that for transaction  $T' \in txns(S^{i+1}) \setminus txns(S^i)$ ,  $T \prec_H^{RT} T'$ . Since  $cseq_i(S^i) = cseq_j(S^j)$  and  $T$  must precede  $T'$  in any serialization, for every  $j > i$ ,  $cseq_j(S^j)[k] = T_m$ .

*(Surjectivity)* The above argument shows that for every  $T \in txns(H)$ , there are  $i, k$ ,  $cseq_i(S^i)[k] = T$ , such that for every  $j > i$ ,  $cseq_j(S^j)[k] = T$ . Thus, for every  $T \in txns(H)$ , there is  $k$  such that  $f(k) = T$ .

*(Injectivity)* Assume, by way of contradiction, that for some  $k \neq m$ ,  $f(k)$  and  $f(m)$  are transactions at indices  $k, m$  of the same  $cseq_i(S^i)$ , and that  $f(k) = f(m)$ . Then  $f(k)$  is the transaction at index  $k$  in some  $cseq_i(S^i)$  and  $f(m)$  is the transaction at index  $m$  in some

$cseq_\ell(S^\ell)$ . For every  $\ell > i$  and  $k < m$ , if  $cseq_i(S^i)[k] = T$ , then  $cseq_\ell(S^\ell)[m] \neq T$  since  $cseq_i(S^i) = cseq_i(S^\ell)$ . If  $\ell > i$  and  $k > m$ , it follows from the definition that  $f(k) \neq f(m)$ . Similar arguments for the case when  $\ell < i$ , imply that  $f(k) \neq f(m)$ .  $\square$

By Claim 6,  $\mathcal{F} = f(1), f(2), \dots, f(i), \dots$  is an infinite sequence of transactions. Let  $S$  be a t-complete t-sequential history such that  $seq(S) = \mathcal{F}$  and for each transaction  $T_k \in txns(H)$ ,  $S|k = H|k$ . Clearly,  $S$  is equivalent to the t-complete history  $H$ .

Let  $\mathcal{F}^i$  be the prefix of  $\mathcal{F}$  of length  $i$ , and  $\hat{S}^i$  be the prefix of  $S$  such that  $seq(\hat{S}^i) = \mathcal{F}^i$ .

**Claim 7.** *Let  $\hat{H}_i^j$  be a subsequence of  $H^j$  reduced to transactions in  $\hat{S}^i$  such that each  $T_k \in txns(\hat{S}^i)$  is t-complete in  $H^j$ . Then, for every  $i$ , there is  $j$  such that  $\hat{S}^i$  is a serialization of  $\hat{H}_i^j$ .*

*Proof.* Let  $H^j$  be the shortest prefix of  $H$  (from  $\mathcal{L}$ ) such that for each  $T \in txns(\hat{S}^i)$ , if  $seq(S^j)[k] = T$ , then for every  $j' > j$ ,  $seq(S^{j'})[k] = T$ . From the construction of  $\mathcal{F}$ , such  $j$  and  $k$  exist. Also, we observe that  $txns(\hat{S}^i) \subseteq txns(S^j)$  and  $\mathcal{F}^i$  is a subsequence of  $seq(S^j)$ . Using arguments similar to the proof of Theorem 1, it follows that  $\hat{S}^i$  is indeed a serialization of  $\hat{H}_i^j$ .  $\square$

Claim 7 completes the proof.  $\square$

From Corollary 2 and Theorem 4, we have:

**Corollary 8.** *Under the restriction that in any infinite history  $H$ , every transaction  $T_k \in txns(H)$  is t-complete in  $H$ , du-opacity is a safety property.*

## 4 Comparison with Other TM Consistency Definitions

In this section, we relate du-opacity with opacity, as defined by Guerraoui and Kapalka [5]. Note that the definition presented in [5] applies to any object with a sequential specification. For the sake of comparison, we restrict it here to TMs with read-write semantics.

**Definition 4** (Guerraoui and Kapalka [4, 5]). *A finite history  $H$  is final-state opaque if there is a legal t-complete t-sequential history  $S$ , such that*

- (1)  $S$  is equivalent to a completion of  $H$  (cf. Definition 2), and
- (2) for any two transactions  $T_k, T_m \in txns(H)$ , if  $T_k \prec_H^{RT} T_m$ , then  $T_k <_S T_m$ .

We say that  $S$  is a final-state serialization of  $H$ .

Figure 3 presents a t-complete sequential history  $H$ , demonstrating that final-state opacity is not a prefix-closed property.  $H$  is final-state opaque, with  $T_1 \cdot T_2$  being a legal t-complete t-sequential history equivalent to  $H$ . Let  $H' = write_1(X, 1), read_2(X)$  be a prefix of  $H$  in which  $T_1$  and  $T_2$  are t-incomplete. By Definition 2,  $T_i$  ( $i = 1, 2$ ) is completed by inserting  $tryC_i \cdot A_i$  immediately after the last event of  $T_i$  in  $H$ . Observe that neither  $T_1 \cdot T_2$  nor  $T_2 \cdot T_1$  are sequences that allow us to derive a serialization of  $H'$  (we assume that the initial value of  $X$  is 0).

A restriction of final-state opacity, which we refer to as *opacity*, was presented in [5] by explicitly filtering out histories that are not prefix-closed.

**Definition 5** (Guerraoui and Kapalka [5]). *A history  $H$  is opaque if and only if every finite prefix  $H'$  of  $H$  (including  $H$  itself if it is finite) is final-state opaque.*

Since final-state opacity is defined only for finite histories, we immediately have:

**Theorem 9.** *Opacity is a safety property.*

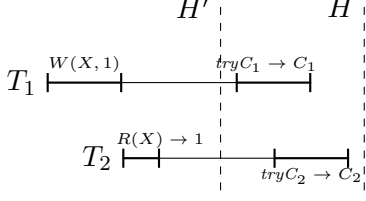


Figure 3: History  $H$  is final-state opaque, while its prefix  $H'$  is not final-state opaque

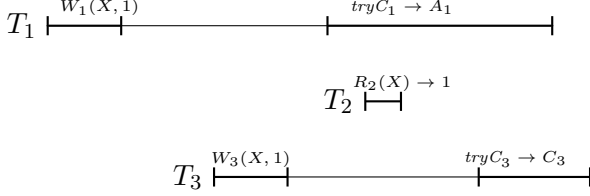


Figure 4: History is opaque, but not du-opaque

#### 4.1 DU-Opacity and Opacity: a separation

**Proposition 2.** *There is an opaque history that is not du-opaque.*

*Proof.* Consider the finite history  $H$  depicted in Figure 4. To prove that  $H$  is opaque, we proceed by examining every prefix of  $H$ .

1. Each prefix up to the invocation of  $read_2(X)$  is trivially final-state opaque.
2. Consider the prefix,  $H^i$  of  $H$  where the  $i^{th}$  event is the response of  $read_2(X)$ . Let  $S^i$  be a t-complete t-sequential history derived from the sequence  $T_1, T_2$  by inserting  $C_1$  immediately after the invocation of  $tryC_1()$ . It is easy to see that  $S^i$  is a final-state serialization of  $H^i$ .
3. Consider the t-complete t-sequential history  $S$  derived from the sequence  $T_1, T_3, T_2$  in which each transaction is t-complete in  $H$ . Clearly,  $S$  is a final-state serialization of  $H$ .

Since  $H$  and every (proper) prefix of it are final-state opaque,  $H$  is opaque.

Consider any possible final-state serialization  $S$  of  $H$ . Since  $T_1$  is aborted in  $H$ ,  $\rho_S(read_2(X)) = T_3$ . But  $read_2(X) \prec_H^{RT} tryC_3()$ —contradiction. Thus,  $H$  is not du-opaque.  $\square$

Note that even under the restriction that every t-operation is sequential, there is an opaque history that is not du-opaque. Figure 5 describes a complete sequential history  $H$  that is opaque, but not du-opaque. No two overlapping transactions in  $H$  write identical values to the same t-object. Observe that every prefix of the history is final-state opaque. But there is only one final-state serialization  $S$  of the history, where  $seq(S) = T_1, T_2, T_3, T_4$  and  $read_4(X)$  reads-from  $T_3$  in  $S$ . Since  $read_4(X) \prec_H^{RT} tryC_3()$ ,  $H$  is not du-opaque.

**Theorem 10.**  $DU\text{-Opacity} \subseteq \text{Opacity}$ .

*Proof.* We first claim that every finite du-opaque history is opaque. Let  $H$  be a finite du-opaque history. By definition, there exists a final-state serialization  $S$  of  $H$ . Since du-opacity is a prefix-closed property, every prefix of  $H$  is final-state opaque. Thus,  $H$  is opaque.

By Corollary 2, every prefix of a du-opaque history is also du-opaque, hence, by Definition 5, every infinite du-opaque history is also opaque.  $\square$

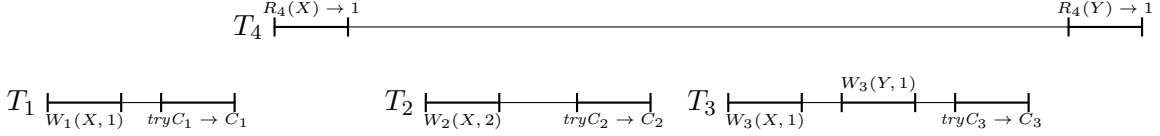


Figure 5: History is opaque, but not du-opaque, even under the overlapping unique writes assumption

Proposition 2 now establishes that du-opacity is indeed a restriction of opacity.

**Corollary 11.**  $DU\text{-}Opacity \subsetneq Opacity$ .

A closer examination of the histories depicted in Figures 4 and 5 reveals interesting insights on the characteristics of opaque histories that are not du-opaque. Consider the history  $H$  depicted in Figure 4. Transaction  $T_1$  invokes  $tryC_1()$  to update the t-object  $X$ , but returns  $A_1$  on observing an overlapping transaction  $T_3$  that also attempts to update  $X$  with the same value. Observe that  $read_2(X)$  reads-from transaction  $T_1$  in any serialization of the prefix  $H^i$  of  $H$  denoting the response of  $read_2(X)$ . However, its read-from transaction shifts to  $T_3$  in any serialization of  $H$ . The history  $\tilde{H}$  depicted in Figure 5 is also interesting because  $T_2$  overwrites the value of  $X$  written by  $T_1$  which subsequently is overwritten by  $T_3$  that writes the same value of  $X$  updated by  $T_1$ . Looking at the prefix  $\tilde{H}^i$  of  $\tilde{H}$ , where the  $i^{th}$  event is the response of  $read_4(X)$ ,  $read_4(X)$  reads-from  $T_1$  in any serialization of  $\tilde{H}^i$ , but reads-from  $T_3$  in any serialization of  $\tilde{H}$ .

We formalize this observation with the following theorem.

**Theorem 12.** *A finite opaque history  $H$  is not du-opaque iff there exists a prefix  $H'$  of  $H$  and some  $read_k(X)$  in  $H'$ , that does not return  $A_k$ , such that for any final-state serialization  $S$  of  $H$ , if  $\rho_S(read_k(X)) = T_m$ , there does not exist any final-state serialization  $S'$  of  $H'$  such that  $\rho_{S'}(read_k(X)) = T_m$ .*

*Proof.* ( $\Rightarrow$ ) Since  $H$  is opaque but not du-opaque, there exists a  $read_k(X)$  in  $H$  such that  $\rho_S(read_k(X)) = T_m$ , but the response  $v$  of  $read_k(X)$  precedes the invocation of  $tryC_m()$  in  $H$ . Now, let  $H'$  denote the prefix of  $H$  of length  $i$  where the  $i^{th}$  event is the response of  $read_k(X)$ . Observe that  $T_m$  is aborted in any final-state serialization  $S'$  of  $H'$ . Thus, there does not exist any  $S'$  such that  $\rho_{S'}(read_k(X)) = T_m$ .

( $\Leftarrow$ ) From Theorem 1, for every du-opaque history  $H$  and any du-opaque serialization  $S$  of  $H$  such that for any prefix  $H'$  of  $H$ , there exists a serialization  $S'$  of  $H'$  such that (1)  $seq(S')$  is a subsequence of  $seq(S)$  and (2) for every  $read_k(X)$  in  $H'$  that does not return  $A_k$ , if  $\rho_S(read_k(X)) = T_m$ ,  $\rho_{S'}(read_k(X)) = T_m$ . Since every du-opaque serialization of a history is also its final-state serialization, the proof follows.  $\square$

## 4.2 DU-Opacity and Opacity: Equivalence

We now show that du-opacity is equivalent to opacity assuming that no two transactions write identical values to the same t-object (“unique-write” assumption).

**Definition 6.** *We say that a property  $\mathcal{P}$  is restricted under unique writes if  $\mathcal{P}$  contains no history  $H$ , in which two transactions  $T_k, T_m \in txns(H)$  perform  $write_k(X, v)$  and  $write_m(X, v)$ , respectively, such that  $v = v'$ .*

The following lemma is now immediate from Theorem 1.

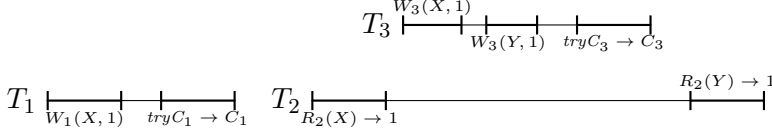


Figure 6: History is du-opaque, but not opaque by the definition in [3]

**Lemma 13.** *Let  $H$  be a finite opaque history restricted to unique writes. Then,  $H$  is du-opaque.*

*Proof.* Let  $H$  be any finite opaque history restricted to unique writes and  $H'$  be any prefix of  $H$ . Let  $S$  be any final-state serialization of  $H$ . It is easy to see that there exists a final-state serialization  $S'$  of  $H'$  such that for every  $read_k(X)$  in  $H'$  that does not return  $A_k$ , if  $\rho_S(read_k(X)) = T_m$ , then  $\rho_{S'}(read_k(X)) = T_m$ . The lemma follows from Theorem 1.  $\square$

We can now prove the following theorem.

**Theorem 14.** *Under the restriction that in any infinite history  $H$ , every transaction  $T_k \in \text{trans}(H)$  is  $t$ -complete in  $H$  and the restriction of unique writes,  $H \in \text{Opacity}$  iff  $H \in \text{DU-Opacity}$ .*

*Proof.* By Definition 5, an infinite history  $H$  is opaque if every finite prefix of  $H$  is final-state opaque. Therefore, Theorem 4 and Lemma 13 imply that  $\text{Opacity} \subseteq \text{DU-Opacity}$ . The converse direction follows from Theorem 10.  $\square$

### 4.3 Other definitions

Explicitly using the deferred-update semantics in an opacity definition was first proposed by Guerraoui et al. [3] and later adopted by Kuznetsov and Ravi [10]. In both papers, opacity is only defined on sequential histories, where every invocation of a  $t$ -operation is immediately followed by a matching response. In particular, these definitions require the serialization to respect the *read-commit order*: if a  $t$ -read of a  $t$ -object  $X$  by a transaction  $T_k$  precedes the  $tryC$  of a transaction  $T_m$  that commits on  $X$ , then  $T_k$  must precede  $T_m$  in the serialization. But we observe that this definition is not equivalent to opacity even for sequential histories: Figure 5 depicts a counter-example. In fact the property defined in [3] is strictly stronger than du-opacity: the history in Figure 6 is du-opaque. We can derive a du-opaque serialization  $S$  for this history such that  $seq(S) = T_1, T_3, T_2$ . However, by the above definition,  $T_2$  must precede  $T_3$  in any serialization of this history since the response of  $R_2(X)$  precedes the invocation of  $tryC_3()$ . In fact, many opaque TM implementations employing read validation within every  $t$ -read or  $tryC$  operation would export such a history.

The recently introduced *TMS2* correctness condition [2, 11] can also be thought of as an attempt to clarify opacity. *TMS2* is a restriction of opacity that explicitly requires the serialization to respect the *read-conflict order* among transactions. Informally, two transactions conflict if they access the same  $t$ -object and at least one of them successfully commits to it. Then, *TMS2* requires that if two transactions  $T_1$  and  $T_2$  conflict on  $t$ -object  $X$  such that  $X \in Wset(T_1) \cap Rset(T_2)$  and  $tryC$  of  $T_1$  precedes the  $tryC$  of  $T_2$ , then  $T_1$  must precede  $T_2$  in any final-state serialization. Since our read-from relation is much less restrictive than the read-conflict order, we suspect that every history that is *TMS2* is du-opaque, but not vice-versa. Indeed, Figure 7 depicts a history  $H$  that is du-opaque, but not *TMS2*. It is du-opaque since there exists a du-opaque serialization  $S$  of  $H$  such that  $seq(S) = T_2, T_1$ . However,  $H$  does not satisfy *TMS2* since  $T_1$  must precede  $T_2$  in any  $t$ -sequential history equivalent to  $H$ , but this is not a final-state serialization of  $H$ .

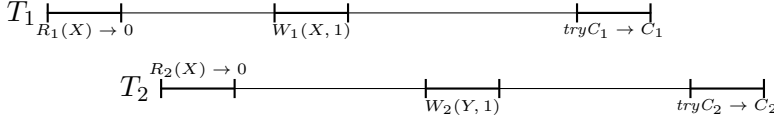


Figure 7: History is du-opaque, but not TMS2 [2]

## 5 Discussion

It is widely accepted that a correctness condition on a set of histories should be a safety property, i.e., should be prefix- and limit-closed. The definition of opacity proposed in [5] forcefully achieves prefix-closure by filtering out prefix-closed histories, and trivially achieves limit-closure by reducing correctness of an infinite history to correctness of its prefixes. In this paper, we proposed a correctness criterion that explicitly disallows reading from an uncommitted transaction, which ensures prefix-closure and (under the restriction that every transaction eventually commits or aborts) limit-closure. We believe that this constructive definition is useful to TM practitioners, since it streamlines possible implementations of t-read and tryC operations. Moreover, it seems that du-opacity already captures the sets of histories exported by most existing opaque TM implementations.

To the best of our knowledge, there is no prior work proving that any TM correctness property is a safety property in the formal sense. The argumentation in the proof of Theorem 4 is inspired by the proof sketch in [12] of the safety of linearizability [8], but turns out to be much trickier due to the complicated “two-layer” structure of opacity. In this paper, we proved that du-opacity is a limit-closed property under the restriction that every transaction is t-complete (has no pending transactions) in the infinite limit history. It remains open whether we can prove limit-closure under the restriction that every transaction is just complete (has no pending t-operations) in the infinite limit history.

**Acknowledgements:** The last author would like to thank Victor Luchangco for interesting discussions on opacity during PODC’12 and anonymous reviewers of WTTM’12 for comments on a nascent version of this paper.

## References

- [1] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, October 1985.
- [2] Simon Doherty, Lindsay Groves, Victor Luchangco, and Mark Moir. Towards formally specifying and verifying transactional memory. *Electron. Notes Theor. Comput. Sci.*, 259:245–261, December 2009.
- [3] Rachid Guerraoui, Thomas A. Henzinger, and Vasu Singh. Permissiveness in transactional memories. In *DISC*, pages 305–319, 2008.
- [4] Rachid Guerraoui and Michal Kapalka. On the correctness of transactional memory. In *PPOPP*, pages 175–184, 2008.
- [5] Rachid Guerraoui and Michal Kapalka. *Principles of Transactional Memory, Synthesis Lectures on Distributed Computing Theory*. Morgan and Claypool, 2010.
- [6] Vassos Hadzilacos. A theory of reliability in database systems. *J. ACM*, 35(1):121–145, 1988.
- [7] Maurice Herlihy and J. Eliot B. Moss. Transactional memory: architectural support for lock-free data structures. *SIGARCH Comput. Archit. News*, 21(2):289–300, 1993.

- [8] Maurice Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.
- [9] Dénes König. *Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe*. Akad. Verlag. 1936.
- [10] Petr Kuznetsov and Srivatsan Ravi. On the cost of concurrency in transactional memory. *CoRR*, abs/1103.1302, 2011.
- [11] Mohsen Lesani, Victor Luchangco, and Mark Moir. Putting opacity in its place. In *WTTM*, 2012.
- [12] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [13] Susan S. Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, 1982.
- [14] Nir Shavit and Dan Touitou. Software transactional memory. In *PODC '95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, pages 204–213, 1995.